

2003 FEB 14 2002

UNITED STATES PATENT APPLICATION

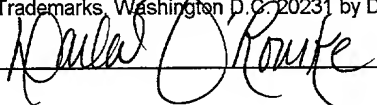
FOR

AN IMPROVED SYSTEM, METHOD AND APPARATUS TO ALLOW  
COMMUNICATION BETWEEN CICS AND NON-CICS SOFTWARE APPLICATIONS

"EXPRESS MAIL" Mailing Label No. EL779651764US

Date of Deposit: January 18, 2002

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington D.C. 20231 by Darla O'Rourke.

  
\_\_\_\_\_

**AN IMPROVED SYSTEM, METHOD AND APPARATUS TO ALLOW  
COMMUNICATION BETWEEN CICS AND NON-CICS SOFTWARE APPLICATIONS**

**Reference to Pending Applications**

This application claims the benefit of U.S. Provisional Application No. 60/262,903, filed January 19, 2001, and incorporated by reference herein in its entirety.

**Reference to Microfiche Appendix**

This application is not referenced in any microfiche appendix.

**Technical Field of the Invention**

In general, the present invention is directed to a middleware tool to allow communication between two or more disparate and distinct computer operating platforms and applications. In particular, the present invention is directed to a middleware tool to facilitate access to, and execution of, existing CICS transactions from a disparate platform requesting application with output from the executed transaction communicated to said requesting application as an XML document.

**Background of the Invention**

The instant invention is a comprehensive software driven solution which provides business enterprises with a quantum leap forward when integrating their existing CICS transactions into scalable e-Business applications. As such the instant invention is believed to be the first solution for invoking existing CICS BMS transactions and delivering their output as an XML document -- not a highly restrictive 3270 screen as provided for in today's prior art systems.

“XML” refers to Extensible Markup Language and allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.

The term “tag” refers to a command inserted in a document that specifies how the document, or a portion of the document, should be formatted. Tags are used by all format specifications that store documents as text files. This includes SGML and HTML.

CICS transactions, running on IBM enterprise servers, are the most common category of host-based e-Business transactions. Enterprises have typically extended these transactions to the web using "web-to-host" products that rely on terminal emulation, screen scraping, proprietary development tools and scripting languages. As corporations continue making the transition from 'bricks-and-mortar' enterprises to 'clicks-and-mortar', they must provide employees, partners, and customers with better access to enterprise data via intranets, extranets and the Internet. This requires a more robust, scalable and cost-effective approach to legacy application and data integration than is afforded by existing art systems. Succinctly stated, industry standards like XML are key to e-Business enablement, but are of no benefit to industry if one of the most common categories of e-Business transactions can not take advantage of its benefit. By replacing terminal-oriented data streams with an XML document containing simple name/value pairs, the instant invention advances the art and drastically reduces the complexity of integrating CICS data with e-Business applications.

“Today, IBM customers use CICS to process over 30 billion transactions per day with a commercial value of several trillion dollars per week,” says Dr. Geoff Sharman, senior consultant with IBM Transaction Systems. “By allowing these customers to XML-enable existing transactions without code changes, the instant invention demonstrates how easily CICS transactions can be

integrated into rapidly deployed and highly scalable e-business applications.” The instant invention facilitates this integration by providing a platform-neutral, XML pathway from CICS transactions to client application irrespective of the computing platform on which they execute. In doing so, the instant invention completely circumvents the use of 3270 data streams and the many, layered components associated with the screen scraping-based solutions of the present art.

"Many enterprises that are using web-to-host solutions which rely on screen scraping techniques are finding they have difficulty scaling," says Darcy Fowkes, research director of Internet services for the Aberdeen Group. "This is unacceptable to organizations that have a large investment in CICS/BMS applications that are running in high volume, production environments." These organizations need an alternative to terminal emulation and screen scraping-based technologies to “e-enable” these production environments. The instant invention provides a more direct, XML pathway to middle-tier application servers and the present invention allows the web-enabling of legacy CICS transactions, replacing web-to-host solutions that rely on screen scraping.

The instant invention expresses output from an existing CICS/BMS transaction as an XML document containing order independent, name-value pairs rather than a fixed format, 3270 screen buffer. This eliminates the client application's dependence on screen formats as well as the need to synchronize changes to the host and client applications. The instant invention is insensitive to changes in the location of fields on a 3270 screen that would cause screen scraping solutions to fail because it intercepts data into, and out of, a CICS transaction before a 3270 data stream is generated as output or expected as input. In fact, using the instant invention, a 3270 data stream is never created. This is possible because the instant invention interacts with CICS applications through the 3270 Bridge interface, a standard feature of CICS Transaction Server version 1.3 (or later). Consequently, no modifications to CICS applications are required; no per-transaction setup is

required either.

The instant invention is compatible with any client application or middle-tier application server that can send a request to an IBM mainframe and receive an XML document. This includes, without limitations, IBM WebSphere™, BEA WebLogic® and SilverStream Application Server. The instant invention also compliments, without limitation, the broad range of evolving middleware technologies based upon XML, including Microsoft BizTalk and SilverStream eXtend™. To paraphrase Fred Holahan, VP & General Manager "Although major application software vendors are embracing XML as their eBusiness interoperability standard, the application portfolios of many enterprises include host-based legacy systems. To succeed in eBusiness integration, enterprises need a range of choices for XML-enabling their host-based applications. The instant invention's technology provides a unique approach for XML-enabling CICS terminal-oriented applications, one that complements SilverStream's legacy integration strategy. Once CICS transactions have been XML-enabled using the instant invention, they can be accessed by eXtend to support real-time integrations with Web portals, trading exchanges, vendor packages and non-CICS legacy applications."

#### **Brief Summary of the Invention**

The present invention is directed to a middleware tool that allows access to, and execution of, existing CICS transactions from a desperate and distinct operating system and application. The present invention embodies a system, method and apparatus to facilitate the invocation of existing CICS transactions and deliver said transactions' output to a requesting application as a standardized XML document via interception of the flow of data into and out of a CICS transaction before a 3270 data stream is generated as output or expected as input. This approach preserves the business logic

of the CICS applications and provides an industry-standard means of exchanging data between CICS and any other XML-enabled applications.

XML is a data format for structured document exchange between applications. Like HTML, it is a markup language derived from SGML. However, unlike HTML, which the Internet community created to format information and display it across multiple platforms, XML is best suited to organize data for structured document exchange between applications. While HTML specifies how a document should appear, it does not describe what kind of information the document contains or how it is organized. XML allows you to organize information in a standard way that can enable disparate systems to conduct business transactions in a known format. For example, business partners can standardize on specific XML syntax to describe purchase orders and can then automate the transfer of that information across otherwise incompatible systems. The key benefit of retrieving host data in XML is that the information is completely independent of how you wish to display it. Because XML is an industry-standard markup language, it can be used for multiple purposes. For example, an eXtensible Style Language Transformation (XSLT) allows you to map XML tags and transform them to any other format.

The present invention runs on the mainframe under OS/390. It is built on the foundation of a feature that IBM has recently added to CICS Transaction Server. The 3270 Bridge makes it possible to intercept the flow of data into, and out of, a CICS transaction *before* a 3270 data stream is generated as output or expected as input. 3270 Bridge works by intercepting the flow of control between the user transaction and BMS, thereby allowing another software component, such as the present invention, to handle input/output operations for the transaction.

The present invention provides a flexible, scalable, and easy-to-use solution that makes CICS applications usable in e-Business by converting application data to XML. Unlike current art solutions

that rely on “screen scraping”, there is no need to identify field locations on a screen. As a result, if CICS developers make changes to their applications by adding or removing fields on a screen, web applications that use the present invention to access the CICS applications will not be affected. This saves time and money for development staff and reduces the chances that users will receive errors when they access your web applications.

Many current host application access solutions run on intermediate UNIX or NT web servers. Running alongside an SNA stack and a web server, these products receive a 3270 datastream, parse it, and then convert the datastream to HTML for presentation in web browsers. Because so much work is done on the single UNIX or NT machine, these products are unable to scale. They are sufficient for simple business-to-consumer (B2C) applications where occasional users logon to check account status or to see if a library book is on the shelves, but when it comes to the needs of B2B applications that produce thousands of transactions each day, traditional web-to-host technology simply cannot keep up.

The present invention is a solution that allows other applications to access legacy data and use it in innovative ways to reduce training costs, improve business efficiency, and unlock the valuable data companies spend years accumulating in their mainframes.

Consequently, it is an object of the instant invention to accept a request from a client program which specifies the name of a CICS transaction and any required input data; cause said transaction to be executed; intercept the input and output commands; and format said output as an XML document to be returned to the client.

A further object of the instant invention is to allow enterprises to integrate existing CICS transactions into scalable e-business applications.

An additional object of the instant invention is to facilitate communication between CICS and non-CICS platforms and applications absent screen scrapping requirements attendant to prior art systems.

A further object of the instant invention is to provide for XML based communication between CICS transactions and client applications absent need for CICS application transaction or system modification.

Other objects and further scope of the applicability of the present invention will become apparent from the detailed description to follow, taken in conjunction with the accompanying drawings wherein like parts are designated by like reference numerals.

### **Description of the Drawings**

Figure 1 illustrates the basic system architecture as well as the present invention's independence from the particular transport mechanism used to exchange requests and response with the client.

Figure 2 illustrates a basic system architecture when utilizing the CICS HTTP Listener as the transport mechanism according to one embodiment of the instant invention.

Figure 3 illustrates a basic system architecture when utilizing either the OS/390 HTTP Server or WebSphere/390 as the transport mechanism according to one embodiment of the instant invention.

Figure 4 illustrates a basic system architecture when utilizing IBM MQSeries as the transport mechanism according to one embodiment of the instant invention.

Figure 5 illustrates a basic system architecture when utilizing Tibco Rendezvous as the transport mechanism according to one embodiment of the instant invention.

Figure 6 illustrates a basic system architecture when utilizing SNA as the transport mechanism according to one embodiment of the instant invention.

Figure 7 illustrates the respective relationships of Figures 7A and 7B and Figures 8 and 8A.

Figure 7A and 7B are flowcharts illustrating executional functionality associated with the invention's software initiation processing component.

Figures 8 and 8A are flowcharts illustrating the functionality of the invention's sub-process for generating an XML document according to an embodiment of the instant invention.

Figure 9 is a flowchart illustrating the functionality of the invention's sub-process for generating an ADS using a Physical Map and ADSD according to an embodiment of the instant

invention.

Figure 10 is a flowchart illustrating the functionality of the invention's sub-process for merging an ADS and the Physical Map according to an embodiment of the instant invention.

5 Figure 11 is a flowchart illustrating the functionality of the invention's sub-process for managing and merging a composite ADS into the current ADS according to an embodiment of the instant invention.

Figures 12 through 18 illustrate the use of a sample CICS BMS transaction from a terminal device.

10 Figures 19 through 22 illustrate the XML output generated by the instant invention during its use with a sample CICS BMS transaction.

### **Detailed Description of the Preferred Embodiment**

While the making and using of various embodiments of the present invention are discussed in detail below, it should be appreciated that the present invention provides for inventive concepts capable of being embodied in a variety of specific contexts. The specific embodiments discussed  
5 herein are merely illustrative of specific manners in which to make and use the invention and are not to be interpreted as limiting the scope of the present invention.

The specification describe the invention presented and the terms that are employed draw their meaning from the use of such terms in the specification. The same terms employed in the prior art may be broader in meaning than specifically employed herein. Whenever there is a question between  
10 the broader definition of such terms used in the prior art and the more specific use of the terms herein, the more specific meaning is meant.

For purposes of full and enabling disclosure, the following code specifications are herein provided to disclose in a comprehensive manner, the teachings and processes wherein the instant invention accepts a client request, invokes the CICS transaction and generates an extensible mark-up  
15 language (XML) document based upon information received from a CICS system via a 3270 Bridge interface.

It is clear, given benefit of the instant invention detail disclosure, that one reasonably skilled in the art could develop similarly intended coding specifications which would replicate the functionality of the processes provided herein. Consequently, it is intended the instant invention  
20 encompasses any such functionally equivalent processes and does not limit itself to this specific coding specifications submitted hereunder.

As used herein, the following provides a glossary of terms referred to in this disclosure and the non-limiting, though generally accepted meanings thereof.

<u>Term</u>	<u>Descriptive Intent</u>
3270 Bridge	See Terminal I/O Intercept Facility
3270 Terminal	A terminal in the IBM 3270 series of terminals (e.g., 3277, 3278, 3279). These terminals communicate with an intermediate terminal controller and understand the 3270 datastream as the means for updating the screen and sending input data back to the controller.
3270 Transaction (or non-BMS Transaction)	A CICS terminal-oriented transaction that does not rely upon BMS to interact with a terminal device. Instead, a 3270 transaction includes programming steps to encode and decode the required 3270 terminal data streams.
ADSD (Application Data Structure Descriptor)	A machine readable collection of information describing the contents of a particular ADS. The ADSD allows a program to correctly interpret the field information within the ADS.
Application Data Structure (ADS)	The machine readable collection of fields associated with a particular map. For each field, the data consists of field value/contents and attributes describing the field.
Basic Mapping Support (BMS)	Basic mapping support (BMS) is an application programming interface between CICS programs and terminal devices. Use of BMS has several advantages. First, BMS removes device dependencies from the application program. It interprets device-independent output commands and generates device-dependent data streams for specific terminals. It also transforms incoming device-dependent data

into device-independent format. These features eliminate the need for programmers to learn complex device data streams. They also allow the programmer to use the same program for a variety of devices, because BMS determines the device information from the terminal definition, not from the application program. Second, BMS separates the design and preparation of screen formats from application logic, reducing the impact of one on the other. Both of these features make it easier to write new programs and to maintain existing code.

(IBM CICS TS 1.3 Application Guide)

BMS Command	BMS transactions invoke the services of BMS by issuing a BMS Commands. The two most common BMS Commands are SEND MAP and RECEIVE MAP.
BMS Transaction	A CICS terminal-oriented transaction that relies upon BMS to interact with a terminal device.
CICS  (Customer Information Control System)	IBM's general-purpose online transaction processing (OLTP) software is an e-business, industrial-strength, server for mission-critical applications. CICS is an application server that runs on S/390 servers and a range of other operating systems, IBM and non-IBM, from the smallest desktop to the largest mainframe. It is used in Client/Server environments and in networks ranging in size from a few terminals to many thousands of terminals. It is a layer of middleware that seamlessly integrates all the basic software services required by OLTP applications together with a rich set of resources and management services in a highly available, reliable, and scaleable manner, enabling its customers to concentrate on the tasks relevant to their particular business. Its application programming interface (API) enables programmers to port applications to and from a wide variety of hardware and software platforms where CICS is available,

and because each product in the CICS family can interface with other members of the CICS family, this enables inter-product communication. Customers may write their own applications or choose from many existing vendor-written products. CICS is an IBM licensed program.

(IBM CICS TS 1.3 Glossary)

Client	The source of a request sent to the instant invention.
Commarea Transaction (or Non-Visual Transaction)	See Non-Visual Transaction.
Context Manager	A repository for storing information to allow the instant invention to serve clients which are stateless and not connected for the same lifetime as the backend resources. Using the context manager, information can be saved between requests from a particular client.
Data Indicator Flag	A flag indicating whether the CICS SEND MAP command specified the MAPONLY option, the DATAONLY option, or neither.
DATAONLY	An option of the CICS SEND MAP command. When present, this option indicates that only the application program data from the ADS is to be displayed on the screen. The contents of the physical map will not be displayed or updated.
Element (XML Element)	See XML.
ERASE	An option of the CICS SEND MAP command. When present, this option causes the terminal screen to be erased before the map and/or data are displayed.

Extended Attribute	Each field within a map may be assigned various extended attributes. These attributes govern such things as color, underlining and highlighting.
Field Attribute	Each field within a map may be assigned various attributes. These attributes govern such things as whether and what the operator can key into the field, whether the cursor stops there, the intensity of the characters, and the initial state of the modified data tag.
HTTP (Hypertext Transfer Protocol)	An application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.
INITIAL	The definition of a field in a BMS map can include the specification of an initial, or default, value. The parameter used to set this value is the INITIAL parameter.
LOAD	A CICS application programming interface command which causes the specified program or mapset to be retrieved from external storage. If successful, the program or mapset requested is then accessible from the program.
Map	In BMS, a format established for a page or a portion of a page, or a set of screen format descriptions. A map relates program variables to the positions in which their values appear on a display device. A map contains other formatting information such as field attributes. A map describes constant fields and their position on the display, the format of input and output fields, the attributes of

constant and variable fields, and the symbolic names of variable fields.

(IBM CICS TS 1.3 Glossary)

The name of a Map must be unique within its Map Set (see below).

#### Map Definition

Definition of the size, shape, position, potential content, and properties of BMS mapsets, maps, and fields within maps, by means of macros.

(IBM CICS TS 1.3 Glossary)

#### Map Generation

The process whereby a Map Definition is accepted as input and a Physical Map and Symbolic Map are generated as output.

#### MAPONLY

An option of the CICS SEND MAP command. When present, this option indicates that only the default data from the physical map is to be displayed on the screen. Application program data will not be displayed or updated.

#### Mapset

In BMS, one or more maps are combined into a Mapset. The effect of this combination reduces the computational overhead associated with managing multiple maps, and allows all maps needed for one application to be referenced and operated on as a unit.

If a map is not combined with other maps into a Mapset, then the map name and Mapset name are the same. The name of a Mapset must be unique within a single CICS system.

#### Non-BMS Transaction

See 3270 Transaction.

(or 3270 Transaction)

**Non-Visual Transaction (or Commarea Transaction)** A transaction designed to be invoked by, and interact with, another program – *not* a human operator at a terminal device. Such a transaction does not emit a visual interface for interaction with a human operator. Instead, the transaction receives input parameters, and returns output parameters, through a mechanism appropriate for program-to-program. By way of example, a CICS “commarea” transaction is a non-visual transaction that receives and returns its input/output data in an area of storage referred to as the “Communication Area”.

**Physical Map** A set of machine-readable instructions and data describing the contents of a map, and how BMS is to display it on the terminal screen.

The Physical Map may optionally include an ADSD. The invention requires that this information exist.

**Query String** The portion of a URL following the ?. A Query String consists of name and value pairs. The name and value are separated by the ‘=’ character. Name/value pairs are then separated by &. A Query String follows a specified format where “unsafe” characters are made “safe” by specifying them using a %xx where the xx is the hexadecimal value of the character in ASCII.

For example: Name1=value1&Name2=value2

**RECEIVE MAP** A CICS application programming interface command which causes BMS to format terminal input data and make it accessible to your application program. The parameters of the RECEIVE MAP command tell BMS:

- Which map to use in formatting the input data stream--that is, what format is on the screen and what data structure the program expects
- Where to find this map (which mapset it resides in)
- Where to get the input
- Whether to suppress translation to upper case
- Where to put the formatted input data

**Request** A collection of parameters and data that instructs the instant invention to execute a particular CICS transaction. The request may contain data to be provided to the transaction to satisfy its initial input requirements.

**RETURN or RETURN IMMEDIATE** A CICS application programming interface command which causes the running transaction to return control to CICS. The IMMEDIATE option causes control to be immediately passed to the program specified by the TRANSID option.

**Root Element** See XML.

**Screen Scraping** “Screen scraping” is a common name/phrase referring to a technique used by programs to integrate with a terminal-oriented program. Screen scraping involves the use of row/column, or linear, coordinates within a program to create a relationship between (or “bind”) data within a screen buffer and program variables. If the location of data on the screen changes (or its relative position within the terminal-oriented data stream), then the binding will be incorrect and the program that relies upon the row/column (or linear) coordinates will usually have to be changed.

One way to determine if screen scraping is taking place within a program process

is to pose the following questions: (1) What happens if the physical location of a data value on the screen is changed?; What happens if the order of a data value within the terminal data stream is changed?; What happens if the location of a literal value on the screen is changed? If a program relies upon screen scraping techniques, various definitional changes will be required before the product will work following such changes.

#### SEND MAP

A CICS application programming interface command which instructs BMS to send mapped output data to a terminal. The parameters of the SEND MAP command tells BMS:

- Which map to use and where to from which mapset
- Where to find the variable data (ADS) for the map and how to merge it with the values from the map (MAPONLY and DATAONLY)
- Which device controls to include in the data stream, and other control options
- Where to put the cursor, if you want to override the position in the map definition

#### Simulated Terminal

A software program which mimics the interface between CICS and a terminal. A simulated terminal would use the Terminal I/O Intercept Facility to capture and respond to input and output commands issued by a transaction.

#### START

A CICS application programming interface command which causes the specified transaction or program to be started.

#### Symbolic Cursor Position

Symbolic cursor positioning is a method for denoting in which field the cursor should be placed when a map is displayed on a terminal screen. Symbolic cursor positioning ties the cursor location to a particular field. The field in which the cursor is to be located is indicated by a flag in the ADS. By tying the cursor

position to a field, as opposed to a particular location on the screen, the field's screen position can be changed and the cursor will still be correctly placed.

**Symbolic Map** A source language data structure that the assembler or compiler uses to resolve source program references to fields within an ADS.  
(IBM CICS TS 1.3 Glossary)

**Tag (XML Tag)** See XML.

**Terminal** In CICS, a device, often equipped with a keyboard and some kind of display, capable of sending and receiving information over a communication channel.

**Terminal I/O Intercept Facility** A mechanism for intercepting the input/output commands and data issued by a CICS transaction. Such a mechanism provides at least the following capabilities: (1) allows a program to request the invocation of a terminal-oriented transaction (as opposed to a human operator at a terminal), and (2) allows a program to intercept and service the terminal I/O requests issued by the transaction *before* the terminal-oriented data stream is generated. The instant invention presumes the existence of such a facility.

By way of example, recent versions of CICS include a terminal I/O intercept facility called "3270 Bridge". 3270 Bridge defines an interface whereby a terminal-oriented transaction can be invoked/operated by another program, rather than a human operator at a terminal. 3270 Bridge allow such a program to intercept BMS commands issued by the transaction before a 3270 terminal data stream is generated. The user transaction runs unchanged as if it were being driven by a real terminal.

**Terminal Identifier** A textual identifier used to designate a specific terminal device used to interact

with CICS.

Terminal-Oriented Transaction    See Visual Transaction.  
(or Visual Transaction)

Token                                A unique value which identifies an entry in the context manager.

Transaction                        A unit of application data processing within CICS (consisting of one or more application programs) initiated by a single request.

Transaction identifier (or    A 4 character identifier which is used to associate a transaction with an  
TRANSID)                        underlying program.

URL (Uniform Resource Locator)    A notation used to represent various entities on a network. The URL consists of constituent parts identifying the network protocol used, host name or IP address, port number and various parameters which have meaning on the machine that hosts the URL. The query string is a portion of the URL and contains instance data in a well formed format.

For example: Protocol://machine:port/resource?query string

Visual Transaction (or Terminal-    A transaction designed for (or that will permit) invocation by, and interaction  
Oriented Transaction)                with, a human operator at a terminal device. As such, the transaction emits a terminal-specific data stream that describes the visual interface used to interact with the human operator. The most common type of terminal that CICS transactions are designed to work with is referred to generically as a “3270” terminal.

The author of a CICS terminal-oriented transaction has two primary alternatives as to how the program will support generation of the terminal-specific data

stream and interaction with the terminal device. The first alternative is for the transaction to include program code that will communicate directly with the terminal device. This implies that the transaction will encode and decode the required terminal-specific data streams. The second alternative is to rely upon an external facility to handle the details associated with terminal interaction. Such an external facility creates a layer of abstraction between the transaction and the terminal device. Basic Mapping Support, a component within CICS, is an example of such a facility.

#### XML (Extensible Markup Language)

XML is a markup language for documents containing structured information and is commonly used for the exchange of data between disparate systems.

Each XML document contains one or more elements, related in a hierarchical manner. The highest level element of an XML document is called the "root" element. All data must be contained by a specific element. XML is referred to as a tagging language because the boundaries of an element are either delimited by start-tags and end-tags, or, for empty elements, by an empty-element tag.

A tag is begun by specifying the name of the tag surrounded by angle brackets. For example, <address>. In this case, "address" is the name of the tag. A tag is ended by including a forward slash character prior to the tag name. For example, </address>. Between the start tag and the end tag would be the value. For example, <address>100 Main Street</address>. If the address element is empty (i.e., has no value), then an empty element tag would be specified as <address/>.

The full XML specification can be found at <http://www.w3.org/TR/2000/REC-xml-20001006>

Figure 1 illustrates the basic system architecture. As shown in Figure 1, the invention's basic components include a Client application, meaning any program or process that can send a request to the present invention, and receive and process the resulting XML document. A Client may be as simple as a web browser, or as complex as an application server, such as IBM WebSphere or BEA WebLogic. Client applications often reside on workstations or servers running the UNIX or Windows operating system.

A typical request/response exchange using the present invention consists of the following steps:

1. The client application sends a request via a transport mechanism to the present invention.
2. The present invention receives the request which indicates the CICS BMS transaction to be invoked or continued along with any input data or parameters.
3. The present invention invokes or continues the requested transaction, indicating to CICS that the present invention will handle all input and output commands issued by the transaction.
4. The CICS BMS transaction executes as though it had been executed from a terminal.
5. All input and output commands issued by the transaction are presented to the present invention for processing.
6. The present invention processes all input and output commands so as to correctly simulate the behavior of a terminal device, and generates an XML document containing the resultant data fields.
7. The present invention returns the XML document back to the client application.

As illustrated in Figure 1, the present invention is neutral with respect to the transport mechanism used by to exchange requests and responses between the client and the instant invention. By way of example, supported transport methods include the following: HTTP using the CICS HTTP Listener; HTTP using the either the OS/390 HTTP Server or WebSphere/390; IBM MQSeries; Tibco Rendezvous, or SNA.

Figure 2 illustrates a basic system architecture when utilizing the CICS HTTP Listener as the transport mechanism. In Figure 2, the following sequence of events results in the retrieval of XML data from CICS.

1. A client application sends an HTTP request with a URL identifying the destination as the CICS host and the present invention as the specific recipient.
2. The HTTP Listener receives the request and relays it to the present invention.
3. The present invention performs as described herein.
4. The present invention passes the resulting XML document to the HTTP Listener.
5. The HTTP Listener send the response to the client.

Figure 3 illustrates a basic system architecture when utilizing either the OS/390 HTTP Server or WebSphere/390 as the transport mechanism. In Figure 3, the following sequence of events results in the retrieval of XML data from CICS.

1. A client application sends an HTTP request with a URL identifying the destination as the CICS host and the present invention as the specific recipient.
2. The OS/390 HTTP Server or WebSphere/390 receive the request and relays it to the present invention.
3. The present invention performs as described herein.

4. The present invention passes the resulting XML document to OS/390 HTTP Server or WebSphere/390.

5. The OS/390 HTTP Server or WebSphere/390 send the response to the client.

Figure 4 illustrates a basic system architecture when utilizing IBM MQSeries as the transport mechanism. In Figure 4, the following sequence of events results in the retrieval of XML data from CICS.

1. A client application places a message on an MQ queue indicating that the intended recipient for the message is the present invention running on the CICS host.
2. IBM MQSeries on the host receive the request and relays it to the present invention.
3. The present invention performs as described herein.
4. The present invention passes the resulting XML document to IBM MQSeries for delivery to the client.
5. IBM MQSeries sends the response to the client.

Figure 5 illustrates a basic system architecture when utilizing Tibco Rendezvous as the transport mechanism. In Figure 5, the following sequence of events results in the retrieval of XML data from CICS.

1. A client application sends a message across the Tibco Rendezvous network indicating that the recipient for the message is the present invention running on the CICS host
2. Tibco Rendezvous on the host receive the request and relays it to the present invention.
3. The present invention performs as described herein.

4. The present invention passes the resulting XML document to Tibco Rendezvous for delivery to the client.

5. Tibco Rendezvous delivers the response to the client.

Figure 6 illustrates a basic system architecture when utilizing SNA as the transport mechanism into the host. This configuration requires a the presence of a middle-tier server running an HTTP server, CICS Transaction Gateway and an SNA protocol stack. The web server can be any HTTP server or application server, like WebSphere, that supports Java servlets. In Figure 6, the following sequence of events results in the retrieval of XML data from CICS.

1. A client application sends an HTTP request with a URL identifying the destination as the web server and the servlet as the specific recipient.
2. The web server receives the request and passes it to the servlet.
3. The servlet invokes the services of CICS Transaction Gateway to forward the request to the present invention across the SNA network.
4. CICS Transaction Gateway used the External Call Interface (ECI) to call the present invention as a subroutine over an SNA connection.
5. The present invention is invoked and performs as described herein.
6. The present invention returns the resulting XML document across the same pathway through which the request was received.
7. The servlet ultimately delivers the response to the client.

Figure 7A and 7B are flowcharts illustrating executional functionality associated with the invention's initiation processing component. Turning now to Figure 7.

In Figure 7, the process is started when an input request is received from a client. The specific form of the input request is unimportant. By way of example, the input request could be expressed as an XML document, or as delimited URL data from an HTTP query string 1. Given that the specific form of the input request may vary, it is interpreted and converted into a standard form: the Request Information structure. The purpose of this structure is to put the request information into a form that is consistent across all requests and utilizes native computer data formats for efficiency 2. The Request Information structure is queried to see whether this is an initial request or a subsequent request. This is determined based upon the presence of a 'token' parameter received with the request. If there is not a token, then it is considered to be an initial request. If there is a valid token, it is considered to be a subsequent request 3. The Context Manager is a data storage device that holds information that is keyed for later retrieval. The Context Manager allows information to be kept independently of the requests. Information stored concerns the context and state of requests. By way of example, the Context Manager is used to save information required to associate a request with CICS resources which may not have the same lifetime 4. A query is next performed using the request's token as the key, which retrieves the context from the prior request 5. For a new request, no context exists in the Context Manager. Generate a unique token for this request. Initialize the context data using the token to identify the new context 6. In order to invoke the requested transaction, the Terminal I/O Intercept Facility needs certain pieces of information. By way of example, the 3270 Bridge requires information such as the transaction identifier, user identification and password credentials, and the input data required to satisfy any initial input commands issued by

2025 RELEASE UNDER E.O. 14176

the transaction. This information also includes parameters which tell 3270 Bridge how the input and output operations are to be intercepted 7. On subsequent requests, data received with the client request is used to satisfy input commands issued by the transaction. By way of example, the most common input command issued by a CICS BMS transaction is RECEIVE MAP. The RECEIVE MAP command expects input data to be presented to it in the form of an ADS. The instant invention must formulate the ADS such that it contains all input data received with the client request and/or required by the application. A 3270 terminal/controller has a data buffer which can be used to store field data between requests. In order to simulate this behavior, the instant invention uses the ADS's and ADSD's received with previous SEND MAP commands to formulate the ADS expected by a RECEIVE MAP command 8. The instant invention requests that the Terminal I/O Intercept Facility begin execution of the requested transaction, and provides it with any required input data or parameters. By way of example, 3270 Bridge requires that the instant invention issue a CICS START command 9. The instant invention requests that the Terminal I/O Intercept Facility continue execution of the requested transaction. The ADS formulated in step #8 is provided to the Terminal I/O Intercept Facility to satisfy the transaction's input command. By way of example, 3270 Bridge requires that the instant invention issue a CICS START command to continue execution of a pseudo-conversational transaction. For a conversational transaction, a CICS START command is not issued 10. The instant invention waits for one of three conditions: (1) the transaction issues an output command; (2) the transaction ends normally or abnormally; (3) the transaction issues an input command which needs to be satisfied 11. When the transaction issues an output command, the instant invention saves the output command and related information (e.g., ADS and ADSD) and returns to step #11 12. Transactions can terminate normally or abnormally. The instant invention



name of the physical map using the MAPSET and MAP options of the SEND MAP command. The physical map is acquired through the use of the CICS LOAD command. Once acquired, the instant invention examines the physical map to determine if the ADSD is present 20. Check to see if Step #20 was successful in retrieving the ADSD from the physical map 21. If the ADSD is successfully retrieved in step #20 it is added to the output command so that it now contains a valid ADSD. The output command now has an ADSD but no ADS. This step calls a procedure that creates an ADS using the physical map and ADSD 22. This step assesses the output command to determine what information the transaction wants to be sent to the terminal (or simulated terminal): either the static data from the physical map, variable data from the ADS, or both. By way of example, a BMS transaction can specify a data indicator flag on a SEND MAP command. Valid values for this flag are MAPONLY and DATAONLY. If MAPONLY is specified, only static data from the physical map will be sent to the terminal. If DATAONLY is specified only variable data from the ADS will be sent to the terminal. If neither option is specified, both static and variable data will be sent to the terminal 22.2. The output command indicated that both the static data from the physical map and the variable data from the ADS are to be sent to the terminal. This step calls a procedure which merges information from the physical map into the ADS 22.4. In order to correctly simulate the behavior of a 3270 terminal, the instant invention must manage the contents of a composite ADS. This step calls a procedure which manages the composite ADS and merges it into the current ADS 22.6. Generate the XML document describing the transaction output. The detailed process is diagramed starting at Step #30 23. The ADSD was not found in the physical map. An XML document is generated which describes the error condition and possible solutions 24. The XML document is sent in response to

the client request received in step #1 28. The instant invention waits for the next request to arrive. Once a request arrives, it begins processing at step #1 29.

Figure 8 and 8A are flowcharts illustrating the functionality of the invention's sub-process for generating an XML document. Turning now to Figure 8. XML requires all documents to have a root element. By way of example, the default root element used by the instant invention is <hostbridge> 31. The status element at this level of the XML document contains information from the instant invention pertaining to its handling of the request. For example, if the instant invention encounters a problem processing the request, that information is conveyed in this element 32. Each output command issued by the transaction is described by an element. The instant invention uses the name of the BMS command issued by the transaction as the name of the element. For example, if a SEND MAP command is issued by the transaction, the element will named SEND\_MAP and started with the tag: <send\_map> 33. The names of the map and mapset, along with the data indicator, specified for this output command are included in the XML document. By way of example, the instant invention generates the following elements to express these values: <mapset>, <map> and <data\_indicator> 33.5. As a result of processing the client request, multiple transactions may have been executed. The results of each transaction must be expressed in the XML document. Each transaction is processed until information regarding all transactions has been included in the XML document 34. The generation of the XML document is now complete. End the root element 37. Each transaction element contains information about a single transaction that was executed. In this section of the XML document, the instant invention includes information such as: transaction identifier; terminal identifier for the simulated terminal; next transaction requested; next transaction requested to run immediately; and the instant invention starts the transaction element with the tag:

<transaction> 38. The status element at this level of the XML document conveys information about the success or failure of the requested transaction. Any error is described using a decimal numeric code as well as a human readable description. The parameters element describes the input parameters received with the client request. The parameters element contains dependent elements which describe the individual parameter names and values specified 40. Return to point from which this procedure was invoked 41. The instant invention ends the transaction element with the tag: </transaction> 45.

The command element of the XML document contains one dependent element for each output command issued by the transaction. This step begins the command element using the tag: <command> 46. For each output command issued by the transaction, the instant invention will perform the steps beginning at Step #33 48. The cursor position has three possible modes: none - no cursor position was specified; positional - a zero based numeric value was specified which represents the offset into the screen buffer of a 3270 terminal; and symbolic Cursor Positioning - one of the fields was flagged as containing the cursor. The cursor type and its associated data are expressed as XML 48.2. The instant invention ends the command element using the tag: </command> 49. Each output command issued by the transaction is described by an element. This step ends the element started in step #33. The instant invention uses the name of the BMS command issued by the transaction as the name of the element. For example, if a SEND MAP command is issued by the transaction, the element will be named send\_map and ended with the tag: </send\_map> 51. The fields element started in step #54 is closed with the end tag: </fields> 53. Start the XML Fields element. By way of example, this element is started by including the <fields> tag in the XML document. The fields element will contain zero or more field elements (one field element for each field in current map) 54. For each field in the current ADS, a field element will be included in the

XML document 55. Different clients may have preferences for how field elements are named. By way of example, the instant invention supports two naming conventions. Field elements can be generated either as <field> or as the actual field name used in the BMS MAP (e.g., <lastname>). The instant invention relies upon a flag, referred to as "Use BMS Field", to determine which convention to use. This flag is controlled by means of an optional parameter provided with the request 57. The "Use BMS Field" flag was not specified. As a result, the instant invention uses a generic identifier for the field element, such as <field> 58. The "Use BMS Field" flag was specified. As a result, the instant invention now generates the field element using the name of the BMS field as the element name. By way of example, for a field named "lastname", the field element would be generated as: <lastname> 59. Since the BMS field name was not used as the name of the field element, a name element is generated. The value of the name element will be the BMS field name. By way of example, for a field named "lastname" the name element would be generated as: <name>lastname</name> 60. The value element is used to contain the current value of a field. By way of example, for a field whose current value is "smith", the value element would be generated as: <value>smith</value>. The attr element is used to specify the current attributes of a field. Field attributes are expressed as attributes of the <attr> element (e.g., <attr byte="60" justify="l" disp="y" prot="y" num="n" int="n" mdt="n" />) 61.

The extattr element is used to specify the current extended attributes of a field. Extended attributes are expressed as attributes of the <extattr> element (e.g., <extattr color="blue" hlt="def" />) 62. Symbolic cursor positioning may be used by a transaction to denote in which field the cursor is to be located. This step queries the current field in the ADS to see if the symbolic cursor positioning flag is on. This flag indicates that the cursor should be placed in this field when displayed

on a 3270 terminal 64. Generate the cursor\_flag element within the current field element. The cursor\_flag element is expressed as a null element (no value); for example, <cursor\_flag/> 65. The tag generated to end the field element depends on how it was started. If the "Use BMS Field" flag was not specified, then the field element will be ended with the tag: </field>. If the "Use BMS Field" flag was specified, the field element will be ended with a tag based upon the name of the current field. By way of example, for a field named "lastname", the field element would be ended with the tag </lastname> 66.

Figure 9 is a flowchart illustrating the process of generating an ADS using a Physical Map and ADSD 70. Using the field information contained in the ADSD, create a corresponding ADS. Set all field values to blank 71. Start loop that processes each field in the newly created blank ADS 72. For the ADS field being processed, locate the corresponding field in the physical map. Note that the format of the ADS and physical map are not the same. For example, the physical map may contain more fields than the ADS 74. The definition of a field in a BMS map can include the specification of an initial/default value (using the INITIAL parameter). For the ADS field being processed, the physical map is queried to see if an INITIAL value was specified for the field 75. For the ADS field being processed, the field's value in the ADS is replaced with the INITIAL field value in the physical map 76. For the ADS field being processed, the field's attribute and extended attributes in the ADS are replaced by the corresponding values in the physical map 77. Return to point from which this procedure was invoked 79.

Figure 10 is a flowchart illustrating the process of merging the ADS and Physical Map 80. Start a loop that processes each field in the ADS 81. For the ADS field being processed, locate the corresponding field in the physical map. Note that the format of the ADS and physical map are not

the same. For example, the physical map may contain more fields than the ADS 82. For the ADS field being processed, compare it's value to binary zeros and branch accordingly 83. For the ADS field being processed, query the corresponding field in the physical map to see if an INITIAL value was defined 84. For the ADS field being processed, replace the field's value in the ADS with the INITIAL field value in the physical map 85. For the ADS field being processed, compare it's field attribute to binary zero and branch accordingly 86. For the ADS field being processed, replace the field's attribute in the ADS with the attribute in the physical map 87. For the ADS field being processed, compare it's extended attributes to binary zeros and branch accordingly 88. For the ADS field being processed, replace the extended attributes in the ADS with the extended attributes in the physical map 89. Return to point from which this procedure was invoked 91.

Figure 11 is a flowchart illustrating the process of managing the composite ADS and merging it into the current ADS 100. A 3270 terminal/controller maintains a buffer which contains the current contents of the screen (all fields). The buffer is cleared whenever a transaction issues an output command with the ERASE option. If a transaction issues multiple output commands (with no intervening ERASE), the buffer will contain the final/composite result from the sequence of output commands. If a particular field is not modified by an output command, then the field's contents remain unchanged. The instant invention simulates this behavior in this procedure by merging the composite ADS into the current ADS (the ADS associated with the output command currently being processed) 100. Check to see if the MAPSET and MAP of the current output request match the MAPSET and MAP of the previous request. If so, instant invention will begin a process to correctly simulate the behavior of a 3270 terminal 101. The output command is queried to determine if the ERASE option was specified. If so, this indicates that the terminal screen should be cleared. If so,

then merging of the previous/composite ADS and current ADS is not needed; the current ADS is used without modification 103. Retrieve the composite ADS from the context manager. 104 The instant invention now merges the composite ADS into the current ADS (the ADS associated with the output command currently being processed). A field value in the composite ADS will be moved into the current ADS if the current ADS does not specify a value for the corresponding field. If the current ADS does contain data for a particular field, then that field's value remains unchanged. Field attributes and extended attributes are also merged in the same manner 105. Store the current ADS in the context manager as the composite ADS. As a result, the current ADS will be used as the composite ADS when processing the next output command. The MAPSET and MAP name associated with the current ADS are also stored. The names are used in step #35 107. End of procedure that manages the composite ADS and merges it into the current ADS 108.

## Illustration of Use

By way of example and instruction, we now illustrate and compare the invocation and use of a sample CICS BMS transaction using both a 3270 terminal and the instant invention.

5           The CICS BMS transaction used in this illustration is one that simulates a stock trading application. The Share Trading Demonstration transaction, or TRAD, guides the end user through a sequence of screens that allows them to first select a company, and then select the action to be performed (i.e., obtain a quote, buy shares, or sell shares). Once these items are specified, it will then display the requested information on the terminal screen. The CICS transaction uses the BMS commands SEND MAP and RECEIVE MAP to communicate with the terminal.

10           First, we will illustrate how to use a 3270 terminal to obtain a stock quote for the Casey Import Export Company using the sample transaction.

15           The first step in executing any CICS transaction from a terminal is entering the transaction name on the screen. The screen image illustrated in Figure 12 (Terminal Input 1) depicts having entered the transaction name TRAD on the screen. Once entered, the end user presses the ENTER key on the keyboard.

          In response to this input, CICS starts the transaction called TRAD. The first operation performed by this transaction is to display the BMS map illustrated in Figure 13 (Terminal Output 1). This screen allows the end user to select the company whose stock is to be acted upon.

20           The transaction now expects the end user to enter a number, 1 thru 4, indicating which company is to be acted upon. The screen image illustrated in Figure 14 (Input 2) depicts having

entered "1" to select the Casey Import Export company. Once entered, the end user presses the ENTER key.

In response, the transaction displays the BMS map illustrated in Figure 15 (Terminal Output 2). This screen allows the end user to select the action to be performed for the selected company.

5 The transaction now expects the end user to enter a number, 1 thru 3, indicating which action is to be performed. The screen image illustrated in Figure 16 (Terminal Input 3) depicts having entered "1" to request a stock quote. Once entered, the end user presses the ENTER key.

In response, the transaction displays the BMS map and data illustrated in Figure 17 (Terminal Output 4). This screen displays the stock quote for the selected company.

10 If this is the last action to be performed by the transaction, it expects the end user to press the PF12 key on the keyboard. This will cause the transaction to terminate. In response to the PF12 key, the transaction displays a message on the terminal screen indicating that it has terminated. This message is illustrated in Figure 18 (Terminal Output 5).

15 The above steps illustrated how an end-user would use a 3270 terminal to invoke and interact with the sample transaction.

We will now illustrate how a client program would invoke and interact with the sample transaction using the instant invention. The sample CICS BMS transaction invoked by the client application via the instant invention is the same as that invoked from the 3270 terminal (no programming changes were made to the sample transaction in order to be used by the instant invention).

20 By way of explanation, the instant invention allows client requests to be articulated as either a structured XML document or as an unstructured command string. In the examples below, client

requests are expressed as a "query string" that could be included with an HTTP request. (An HTTP query string begins with a "?", and individual parameters are separated by an "&"). By way of illustration, parameters that begin with the characters "req\_" will be interpreted by the instant invention as information required to control the processing of the request. Parameters that do not  
5 begin with these characters will be interpreted as transaction variables.

Requests may be sent to the instant invention using various methods. It is important to note that the functionality of the instant invention does not depend, nor change, based upon either the method selected to express or exchange requests and responses with the client.

The first request that must be sent from the client to the instant invention is one that specifies the name of the transaction to be invoked. To invoke the TRAD transaction, the following request might be used:

```
?req_tranid=trad&req_aid=enter
```

This request instructs the instant invention to start a transaction named "trad". In response, the instant invention would respond with the XML document illustrated in Figure 19. This document corresponds to Figure 13 (Terminal Output 1).

Note that the XML document illustrated in Figure 19 includes a <token> element indicating that the transaction specified a "next transaction id". The client must return this token to the instant invention on subsequent requests so that the instant invention can determine whether this is an "initial" or "subsequent" request (as described in the flowchart).

As before, the transaction now expects the client to select the company to be acted upon. To do so, the client would send the following request to the instant invention:

```
?req_tranid=trad&req_token=ca445fb5&req_aid=enter&option=1
```

This request instructs the instant invention to continue the previous transaction and pass to it the value of "1" for the variable named "option". In response, the instant invention would respond with the XML document illustrated in Figure 20. This document corresponds to Figure 15 (Terminal Output 2).

5           The transaction now expects the client to select a number, 1 thru 3, indicating which action is to be performed. As before, a stock quote will be requested. To do so, the client would send the following request to the instant invention:

```
?req_tranid=trad&req_token=ca445fb5&req_aid=enter&opt2=1
```

10           This request instructs the instant invention to continue the previous transaction and pass to it the value of "1" for the variable named "opt2". In response, the instant invention would respond with the XML document illustrated in Figure 21. This document corresponds to Figure 17 (Terminal Output 3).

15           Upon receipt of XML Response 3, the client has obtained the desired information. As before, the transaction will now be terminated. To do so, the client would send the following request to the instant invention:

```
?req_tranid=trad&req_token=ca445fb5&req_aid=pf12
```

20           This request instructs the instant invention to continue the previous transaction, passing to it the indication that it should terminate. In response, the instant invention would respond with the XML document illustrated in Figure 22, indicating that the transaction has terminated. Note that this final XML document does not include a <token> element. This indicates that the transaction has ended without specifying a "next transaction id".

20670-4300

5

The claims and the specification describe the invention presented and the terms that are employed in the claims draw their meaning from the use of such terms in the specification. The same terms employed in the prior art may be broader in meaning than specifically employed herein. Whenever there is a question between the broader definition of such terms used in the prior art and the more specific use of the terms herein, the more specific meaning is meant.

10

While the invention has been described with a certain degree of particularity, it will be recognized by those skilled in the art that many changes may be made. In the sequence of process step execution, details of construction and the arrangement of components without departing from the spirit and scope of this disclosure. It is understood that the invention is not limited to the embodiments set forth herein for purposes of exemplification, but is to be limited only by the scope of the attached claim or claims, including the full range of equivalency to which each element thereof is entitled.

15

The following code specification was provided in U.S. Provisional Application No. 60/262,903, filed January 19, 2001, and is incorporated in its entirety herein. As such, the code specification provides for complete though non-limiting disclosure of the process used by the present invention to generate an XML document. This specification is further illustrated by the flowchart in Figure 8A and 8B.

A sample XML document structure created by this process is also illustrated.

The following conventions are used within this document:

20

"BEGIN TAG:" means that a beginning tag, of the indicated name, will be included in the XML document.

"END TAG:" means that an ending tag, of the indicated name, will be included in the XML document.

"INCLUDE:" means that the specified information or value will be included in the XML document.

5 "PERFORM:" means that the specified process is performed at this point.

#### Begin Process: Generate\_XML\_Output

INCLUDE: Standard XML header

INCLUDE: XML comment with HostBridge copyright information

10 BEGIN TAG: HostBridge

BEGIN TAG: token

INCLUDE: HostBridge token value

END TAG: token

\*\*\* NOTE: Refer to XML Output: Point A

15 BEGIN TAG: status

BEGIN TAG: response

INCLUDE: response value

END TAG: response

BEGIN TAG: description

20 INCLUDE: response description

END TAG: description

BEGIN TAG: cics\_response

INCLUDE: cics\_response value

END TAG: cics\_response

25 BEGIN TAG: cics\_description

5

20

25

INCLUDE: cics\_description value

END TAG: cics\_description

BEGIN TAG: cics\_response2

INCLUDE: cics\_response2 value

END TAG: cics\_response2

END TAG: status

\*\*\* NOTE: Refer to XML Output: Point B

BEGIN TAG: transaction

BEGIN TAG: facility\_name

INCLUDE: facility\_name value

END TAG: facility\_name

BEGIN TAG: facility\_keep\_time

INCLUDE: facility\_keep\_time value

END TAG: facility\_keep\_time

\*\*\* NOTE: Refer to XML Output: Point C

PERFORM: Include\_Transaction\_Parameters

\*\*\* NOTE: Refer to XML Output: Point D

PERFORM: Include\_Transaction\_Data

\*\*\* NOTE: Refer to XML Output: Point E

END TAG: transaction

END TAG: HostBridge

\*\*\* NOTE: Refer to XML Output: Final

End Process: Generate\_XML\_Output

Begin Process: Include\_Transaction\_Parameters

BEGIN TAG: parameters

5

15

20

25

BEGIN TAG: input

BEGIN TAG: tranid

INCLUDE: tranid value

END TAG: tranid

BEGIN TAG: aid

INCLUDE: aid value

END TAG: aid

BEGIN TAG: cursor\_position

INCLUDE: cursor\_position value

END TAG: cursor\_position

END TAG: input

BEGIN TAG: output

BEGIN TAG: task\_end\_status

INCLUDE: task\_end\_status value

END TAG: task\_end\_status

BEGIN TAG: abend\_code

INCLUDE: abend\_code value

END TAG: abend\_code

BEGIN TAG: next\_tranid

INCLUDE: next\_tranid value

END TAG: next\_tranid

BEGIN TAG: next\_start\_code

INCLUDE: next\_start\_code value

END TAG: next\_start\_code

END TAG: outout

END TAG: parameters

End Process: Include\_Transaction\_Parameters

Begin Process: Include\_Transaction\_Data

BEGIN TAG: command

For each command in the input data

{

\*\*\* NOTE: At this point, we generate the XML to describe the specific commands included in the input data. A number of different commands are possible. The following process illustrates the steps taken to process the "Send Map" command. This command is the most complex. Processing of all other commands are accomplished using a subset of the "Send Map" process.

PERFORM: Include\_Send\_Map\_Command

}

END TAG: command

End Process: Include Transaction Data

Begin Process: Include\_Send\_Map\_Command

BEGIN TAG: send\_map

PERFORM: Include\_Control\_Indicators

BEGIN TAG: mapset

INCLUDE: mapset value

END TAG: mapset

BEGIN TAG: map

INCLUDE: map value

END TAG: map

BEGIN TAG: data\_indicator

INCLUDE: data\_indicator value  
END TAG: data\_indicator  
PERFORM: Include\_Data\_Fields

END TAG: send\_map

5 End Process: Include\_Send\_Map\_Command

Begin Process: Include Control Indicators

BEGIN TAG: control\_indicators

BEGIN TAG: erase

INCLUDE: erase value

END TAG: erase

BEGIN TAG: erase\_all\_unprot

INCLUDE: erase\_all\_unprot value

END TAG: erase\_all\_unprot

BEGIN TAG: unlock\_kbd

INCLUDE: unlock\_kbd value

END TAG: unlock\_kbd

BEGIN TAG: alarm

INCLUDE: alarm value

END TAG: alarm

BEGIN TAG: reset\_kbd

INCLUDE: reset\_kbd value

END TAG: reset\_kbd

BEGIN TAG: last\_output

INCLUDE: last\_output value

END TAG: last\_output

BEGIN TAG: cursor

INCLUDE: cursor value

END TAG: cursor

END TAG: control\_indicators

End Process: Include\_Control\_Indicators

5 Begin Process: Include\_Data\_Fields

BEGIN TAG: data\_descriptor

BEGIN TAG: field\_count

INCLUDE: field\_count value

END TAG: field\_count

BEGIN TAG: attribute\_number

INCLUDE: attribute\_number value

END TAG: attribute\_number

BEGIN TAG: attribute\_type\_codes

INCLUDE: attribute\_tupe\_codes value

END TAG: attribute\_type\_codes

BEGIN TAG: write\_control\_char

INCLUDE: write\_control\_char value

END TAG: write\_control\_char

END TAG: data\_descriptor

20 BEGIN TAG: data

For each data field in the input data

{

BEGIN TAG: field

BEGIN TAG: name

25 INCLUDE: name value

END TAG: name

5

20

25

BEGIN TAG: value

INCLUDE: value value

END TAG: value

BEGIN TAG: length

INCLUDE: length value

END TAG: length

END TAG: field

}

END TAG: data

End Process: Include\_Data\_Fields

XML Output: Point A

<?xml version="1.0" ?>

<!-- HostBridge Generated Output -->

<!-- HostBridge Copyright (c) 2000, 2001 Russell W. Teubner -->

<HostBridge>

<token> \_\_\_\_ </token>

XML Output: Point B

<?xml version="1.0" ?>

<!-- HostBridge Generated Output -->

<!-- HostBridge Copyright (c) 2000, 2001 Russell W. Teubner -->

<HostBridge>

<token> \_\_\_\_ </token>

<status>

<response> \_\_\_\_ </response>

```

    <description>____</description>

    <cics_response>____</cics_response>

    <cics_description>____</cics_description>

    <cics_response2>____</cics_response2>

```

5           </status>

## XML Output: Point C

```

<?xml version="1.0" ?>

<!-- HostBridge Generated Output -->

<!-- HostBridge Copyright (c) 2000, 2001 Russell W. Teubner -->

<HostBridge>

    <token>____</token>

    <status>

        <response>____</response>

        <description>____</description>

        <cics_response>____</cics_response>

        <cics_description>____</cics_description>

        <cics_response2>____</cics_response2>

    </status>

    <transaction>

```

20           <facility\_name>\_\_\_\_</facility\_name>

              <facility\_keep\_time>\_\_\_\_</facility\_keep\_time>

## XML Output: Point D

```

<?xml version="1.0" ?>

<!-- HostBridge Generated Output -->

<!-- HostBridge Copyright (c) 2000, 2001 Russell W. Teubner -->

<HostBridge>

```

<token>\_\_\_\_</token>

<status>

<response>\_\_\_\_</response>

<description>\_\_\_\_</description>

<cics\_response>\_\_\_\_</cics\_response>

<cics\_description>\_\_\_\_</cics\_description>

<cics\_response2>\_\_\_\_</cics\_response2>

</status>

<transaction>

<facility\_name>\_\_\_\_</facility\_name>

<facility\_keep\_time>\_\_\_\_</facility\_keep\_time>

<parameters>

<input>

<tranid>\_\_\_\_</tranid>

<aid>\_\_\_\_</aid>

<cursor\_position>\_\_\_\_</cursor\_position>

</input>

<output>

<task\_end\_status>\_\_\_\_</task\_end\_status>

<abend\_code>\_\_\_\_</abend\_code>

<next\_tranid>\_\_\_\_</next\_tranid>

<next\_start\_code>\_\_\_\_</next\_start\_code>

</output>

</parameters>

XML Output: Point E

<?xml version="1.0" ?>

<!-- HostBridge Generated Output -->

<!-- HostBridge Copyright (c) 2000, 2001 Russell W. Teubner -->

<HostBridge>

<token>\_\_\_\_</token>

<status>

<response>\_\_\_\_</response>

<description>\_\_\_\_</description>

<cics\_response>\_\_\_\_</cics\_response>

<cics\_description>\_\_\_\_</cics\_description>

<cics\_response2>\_\_\_\_</cics\_response2>

</status>

<transaction>

<facility\_name>\_\_\_\_</facility\_name>

<facility\_keep\_time>\_\_\_\_</facility\_keep\_time>

<parameters>

<input>

<tranid>\_\_\_\_</tranid>

<aid>\_\_\_\_</aid>

<cursor\_position>\_\_\_\_</cursor\_position>

</input>

<output>

<task\_end\_status>\_\_\_\_</task\_end\_status>

<abend\_code>\_\_\_\_</abend\_code>

<next\_tranid>\_\_\_\_</next\_tranid>

<next\_start\_code>\_\_\_\_</next\_start\_code>

</output>

</parameters>

<command>

<send\_map>

<control\_indicators>

<erase>\_\_\_\_</erase>

<erase\_all\_unprot>\_\_\_\_</erase\_all\_unprot>

<unlock\_kbd>\_\_\_\_</unlock\_kbd>

<alarm>\_\_\_\_</alarm>

<reset\_kbd>\_\_\_\_</reset\_kbd>

<last\_output>\_\_\_\_</last\_output>

<cursor>\_\_\_\_</cursor>

</control\_indicators>

<mapset>\_\_\_\_</mapset>

<map>\_\_\_\_</map>

<data\_indicator>\_\_\_\_</data\_indicator>

<data\_descriptor>

<field\_count>\_\_\_\_</field\_count>

<attribute\_number>\_\_\_\_</attribute\_number>

<attribute\_type\_codes>\_\_\_\_</attribute\_type\_codes>

<write\_control\_char>\_\_\_\_</write\_control\_char>

</data\_descriptor>

<data>

<field>

<name>\_\_\_\_</name>

<value>\_\_\_\_</value>

```

        <length>____</length>

    </field>

    .
    .
    . *** NOTE: Occurs multiple times based upon the
    . number of fields in the input data.
    .

```

```

    <field>

        <name>____</name>

        <value>____</value>

        <length>____</length>

    </field>

```

```

        </data>

    </send_map>

</command>

```

XML Output: FINAL

```

<?xml version="1.0" ?>

<!-- HostBridge Generated Output -->

<!-- HostBridge Copyright (c) 2000, 2001 Russell W. Teubner -->

<HostBridge>

    <token>____</token>

    <status>

        <response>____</response>

        <description>____</description>

        <cics_response>____</cics_response>

        <cics_description>____</cics_description>

        <cics_response2>____</cics_response2>

```

</status>

<transaction>

<facility\_name>\_\_\_\_</facility\_name>

<facility\_keep\_time>\_\_\_\_</facility\_keep\_time>

<parameters>

<input>

<tranid>\_\_\_\_</tranid>

<aid>\_\_\_\_</aid>

<cursor\_position>\_\_\_\_</cursor\_position>

</input>

<output>

<task\_end\_status>\_\_\_\_</task\_end\_status>

<abend\_code>\_\_\_\_</abend\_code>

<next\_tranid>\_\_\_\_</next\_tranid>

<next\_start\_code>\_\_\_\_</next\_start\_code>

</output>

</parameters>

<command>

<send\_map>

<control\_indicators>

<erase>\_\_\_\_</erase>

<erase\_all\_unprot>\_\_\_\_</erase\_all\_unprot>

<unlock\_kbd>\_\_\_\_</unlock\_kbd>

<alarm>\_\_\_\_</alarm>

<reset\_kbd>\_\_\_\_</reset\_kbd>

<last\_output>\_\_\_\_</last\_output>

<cursor> \_\_\_\_ </cursor>

</control\_indicators>

<mapset> \_\_\_\_ </mapset>

<map> \_\_\_\_ </map>

<data\_indicator> \_\_\_\_ </data\_indicator>

<data\_descriptor>

<field\_count> \_\_\_\_ </field\_count>

<attribute\_number> \_\_\_\_ </attribute\_number>

<attribute\_type\_codes> \_\_\_\_ </attribute\_type\_codes>

<write\_control\_char> \_\_\_\_ </write\_control\_char>

</data\_descriptor>

<data>

<field>

<name> \_\_\_\_ </name>

<value> \_\_\_\_ </value>

<length> \_\_\_\_ </length>

</field>

.

. \*\*\* NOTE: Occurs multiple times based upon the

. number of fields in the input data.

.

<field>

<name> \_\_\_\_ </name>

<value> \_\_\_\_ </value>

<length> \_\_\_\_ </length>

</field>

</data>

</send\_map>

</command>

</transaction>

5 </HostBridge>